### Shell脚本编程

* 脚本：可以运行的一个文件，并且可以实现某种功能（命令的堆积）
* 规范shell脚本的一般组成

1. 环境声明（sha-bang）

下面所有的语句解释程序

2.注释文本

3.可执行代码

* 重定向输出
  + - 1. >: 只是把前面命令正确的输出结果重定向
      2. 2>:只是把前面命令错误的输出结果重定向
      3. &>:前面命令正确与错误的结果重定向
      4. &2：将前面命令的结果性质改变为错误的输出

[root@server0 ~]# echo 123 > /opt/1.txt

[root@server0 ~]# cat /opt/1.txt

[root@server0 ~]# cat /opt/1.txt /etc/

[root@server0 ~]# cat /opt/1.txt /etc/ > /opt/a.txt

[root@server0 ~]# cat /opt/1.txt /etc/ 2> /opt/a.txt

[root@server0 ~]# cat /opt/1.txt /etc/ &> /opt/a.txt

/dev/null:黑洞设备

* 变量：容器以不变的名称，存储可以变化的值（作用：让脚本适用多变的环境）$:调用变量
* 为变量赋值：

1.变量名=值 （登号的两边不允许有空格）

[root@server0 ~]# vim /root/user.sh

#!/bin/bash

user=harry

useradd $user &> /dev/null

echo 用户$user创建成功

echo 123 | passwd --stdin $user &> /dev/null

echo 用户$user密码设置成功

[root@server0 ~]# /root/user.sh

2.read 为变量赋值，读入用户在键盘上的输入，将用户在键盘上的输入赋值给变量

#!/bin/bash

read -p '请输入你想要创建的用户：' user

（-p:输出提示信息）

read -p ‘请输入你想要设置的密码：’ pass

useradd $user &> /dev/null

echo 用户$user创建成功

echo $pass | passwd --stdin $user &> /dev/null

echo 用户$user密码设置成功

* 设置变量时的注意事项：
* 等号两边不允许有空格
* 变量名由字母/数字/下划线组成，区分大小写
* 变量名不能以数字开头，不要用关键字和特殊字符

查看引用变量

$[]:代表运算

[root@server0 ~]# echo $[2\*3]

[root@server0 ~]# a=7

[root@server0 ~]# b=3

[root@server0 ~]# echo $[$a+$b]

$()或者``：将命令的输出结果，作为参数

[root@server0 opt]# mkdir `date +%F`

[root@server0 opt]# ls

[root@server0 opt]# mkdir mariadb-`date +%F`

[root@server0 opt]# ls

[root@server0 opt]# mkdir `hostname`-`date +%F`

单引号’’:可以取消特殊字符的意义

$RANDOM:生成随机数

生成随机数范围为0~99之间的随机数 :$[$RANDOM%100]

生成随机数范围为0~9之间的随机数 :$[$RANDOM%10]

利用的是求余运算：余数一定小于被除数

变量的种类

一、环境变量：由系统直接定义完成，赋值完成

常见的环境变量：USER ：当前登陆的系统身份

二、位置变量：在执行脚本的时候提供命令行的参数

表示为$n,n为序号

$1、$2、……${10}$、{11}

[root@server0 ~]# vim /root/test02.sh

[root@server0 ~]# cat /root/test02.sh

#! /bin/bash

cat -n $1 |head -$2

[root@server0 ~]# chmod +x /root/test02.sh

[root@server0 ~]# /root/test02.sh /etc/passwd 4

/etc/passwd 代表$1 4代表$2

1. 预定义变量：用来保存脚本程序的执行信息
   * + 1. $#:以加载的位置变量的个数（计算使用变量的总数和）
       2. $\*:显示所有位置变量的值
       3. $?:程序退出后的状态值，0表示正常，1表示不正常

储存上一个命令的返回值

4. $!:最后一个放入后台的进程号，只与&有关

[root@server0 ~]# vim /root/test02.sh

#! /bin/bash

cat -n $1 |head -$2

echo $#

echo $\*

[root@server0 ~]# /root/test02.sh /etc/passwd 2

1 root:x:0:0:root:/root:/bin/bash

2 bin:x:1:1:bin:/bin:/sbin/nologin

2

/etc/passwd 2

条件测试及选择

检查文件状态：

-e：存在为真

[root@server0 ~]# [ -e /etc ] 中括号内每一部分都要有空格

[root@server0 ~]# echo $?

-d：存在且为目录 才为真

-f：存在且为文件 才为真

-w:存在且对于对象有写入操作 才为真

-r:存在且对于对象有读操作 才为真

-x:存在且对于对象有执行操作 才为真

比较整数大小（带g的都有大于二字，带e的都有等于二字，带l的都有小于二字）

-gt：大于 [root@server0 ~]# [ 2 -gt 1 ]

[root@server0 ~]# echo $?

[root@server0 ~]# a=7

[root@server0 ~]# b=3

[root@server0 ~]# [ $a -ne $b ]

[root@server0 ~]# echo $？

[root@server0 ~]# [ $a -ge $b ]

[root@server0 ~]# echo $?

[root@server0 ~]# [ $a -le $b ]

-ge：大于等于

-eq：等于

-ne；不等于

-lt：小于

-le：小于等于

字符串的比对

==：字符串一致为真

！=：字符串不一致为真

[root@server0 ~]# [ $USER == root ]

[root@server0 ~]# echo $?

[root@server0 ~]# [ redhat == root ]

[root@server0 ~]# echo $?

[root@server0 ~]# [ redhat != root ]

[root@server0 ~]# echo $?

### If选择结构

if双分支处理：

if 条件判断 ； then

命令序列01

else

命令序列02

fi

* 案例：计算机随机产生一个0-9之间的数字

用户输入一个0-9之间的数字

#! /bin/bash

num1=$[$RANDOM%10]

read -p '请输入一个0-9之间的数字：' num2

if [ $num1 -eq $num2 ];then

echo 恭喜您猜对了！

else

echo 谢谢参与！

fi

if 多分支处理：

if [条件判断1]；then

命令序列01

elif [条件判断2]；then

命令序列02

elif [条件判断3]；then

命令序列03

elif [条件判断4]；then

命令序列04

else

命令序列05

fi

#!/bin/bash

read -p '请输入您的成绩：' nsd

if [ $nsd -ge 90 ];then

echo 优秀

elif [ $nsd -ge 80 ];then

echo 良好

elif [ $nsd -ge 70 ];then

echo 及格

elif [ $nsd -ge 60 ];then

echo 仍需努力

else

echo 在牛的肖邦，也弹不出哥的悲伤

fi

exit : 退出脚本

http://blog.csdn.net/boshuzhang/article/details/51122457

### For循环结构

变量a参与循环，变量a值影响循环的结果

[root@server0 ~]# vim /root/for.sh

[root@server0 ~]# chmod +x /root/for.sh

#! /bin/bash

for a in nsd11 nsd12 nsd13 nsd15

do

useradd $a

echo $a 创建成功

done

变量a参与循环，变量a值影响循环的结果

#!/bin/bash

num1=$[$RANDOM%10]

for a in 1 2 3

do

read -p '请输入0-9之间的数字：' num2

if [ $num2 -eq $num1 ];then

echo true

elif [ $num2 -gt $num1 ];then

echo big

else

echo little

fi

done

造数工具： {1..100}

拓展：#!/bin/bash

for b in {1..1000}

do

read -p '请输入你的成绩：' a

if [ $a -gt 100 ] 2>/dev/null;then

echo li害 && exit

else

if [ $a -gt 0 ] 2>/dev/null;then

if [ $a -ge 90 ];then

echo 优秀

elif [ $a -ge 80 ];then

echo 良好

elif [ $a -ge 70 ];then

echo 及格

elif [ $a -ge 60 ];then

echo 仍需努力

else

echo 再见

fi

else

echo 请输入正确的成绩

fi

fi

Done

* 两个随机数比大小

#!/bin/bash

for a in {1..1000}

do

num1=$[$RANDOM%100]

num2=$[$RANDOM%100]

if [ $num1 -gt $num2 ];then

echo $num1 '> ' $num2

elif [ $num1 -eq $num2 ];then

echo $num1 '= ' $num2

else

echo $num1 '< ' $num2

fi

done

* 脚本案例：

输入一个用户名，判断该用户是否存在，

如果存在则输出该用户以存在

如果不存在则创建该用户

#!/bin/bash

read -p '请输入用户名' user

id $user &>/dev/null

if [ $? -eq 0 ];then

echo 用户以存在

else

echo 用户不存在

useradd $user

echo 用户$user 创建成功

fi

* 输入一个ip，判断本机是否可以与ip通信，

#!/bin/bash

read -p '请输入ip地址：' user

ping -c 2 $user &>/dev/null

if [ $? -eq 0 ];then

echo 可以通信

else

echo 不可以通信

for c in {1..100}

do

read -p '是否再次输入ip地址（yes/no）：' a

if [ $a == yes ];then

read -p '请再次输入ip地址：' b

ping -c 2 $b &>/dev/null

if [ $? -eq 0 ];then

echo 可以通信 && exit

else

echo 不可以通信

fi

elif [ $a == no ];then

exit

else

echo 请输入yes/no

fi

done

fi

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#!/bin/bash

if [ $# -ge 0 ];then #判断位置变量的个数是否为0

echo '/root/foo.sh redhat|fedora' >&2

# >&2:表示将正确的输出结果变成错误的输出

exit 1

#脚本退出的返回值

else

if [ $1 == redhat ];then

echo fedora

elif [ $1 == fedora ];then

echo redhat

else

echo '/root/foo.sh redhat|fedora' >&2

exit 1

fi

fi

#!/bin/bash

if [ $# -eq 0 ];then

echo 'Usage:/root/batchusers' >&2

exit 1

if [ -f $1 ];then

for i in 'cat $1'

do

useradd -s /bin/false $i

echo $i 成功

done

else

echo 'Input file not found' >&2

exit1

fi

fi

A && B:A成功时，才执行B

A || B：A失败时才执行B

A ； B：执行A 执行B

判断目录是否存在，存在就将光盘挂载在目录，不存在就创建目录，挂载光盘

[ -d /media/abc ] || mkdir /media/abc

[ ! -d /media/abc ] && mkdir /media/abc